PROGRESS REPORT NUMBER 10

of the

Research and Educational Activities

in

Machine Computation

by the

Cooperating Colleges of New England

January 1962

Computation Center
Massachusetts Institute of Technology
Cambridge 39, Massachusetts

I. RESEARCH ACTIVITIES

1.1 An Experimental Time-Sharing System for the IBM 709 Computer

F.J. Corbató
M.L. Merwin
R.C. Daley

During the past six months, the first version of the time-sharing Supervisor for the 709 was completed. The background system was implemented so that Fortran as well as Mad and Fap inputs were acceptable. The Fortran input is converted to Mad via Madtran and compiled with Mad. The foreground system was expanded so that Fap and Mad programs could be input, edited, translated, loaded, and queried through a post-mortem program. The full system was demonstrated at a Computation Center seminar in November with a realistic set of jobs running as a background program and with varied jobs being performed at each of the two Flexowriters. The final checkout of the system so closely coincided with the removal of the 709 on December 1 that there has been no chance to gain any operating experience on a field test level.

The programming and coding to accommodate hardware changes in the rack between the 709 and 7090 have been completed. Some modifications, mostly simplifications, have been made in the Supervisor owing to the differing behavior of the 709 and 7090 under trapping conditions. The programming for effective usage of the relocation and protection modes on the 7090 has been started.

The following is a brief, user-oriented description of an experimental time-sharing system that was designed for use with the 709 computer, two Flexowriter typewriters, and the real-time equipment rack designed and built under the direction of H. Teager and his group. The time-sharing occurs between three users, two of whom are on-line each at a typewriter in a foreground system, and a third passive user of the background Fap-Mad-Madtran BSS Monitor System (see Memo CC-177) similar to the FMS monitor system used by most of the Center programmers.

Significant features of the foreground system are that it allows the user to: 1) develop programs in languages compatible with the background system, 2) develop a private file of programs, 3) start debugging sessions at the state of the previous session, and 4) set his own pace with little waste of computer time. Core storage is allocated such that all users operate in the upper 27K with the time-sharing Supervisor (TSS) permanently in the lower 5K. To avoid memory allocation clashes, protect users from one another, and simplify the initial system organization, only one user has been kept in core memory at a time. User swaps are kept down to overlapped magnetic tape dump/undumps of the pertinent locations in the two user programs.

The foreground system is organized around commands that the user can give on his typewriter and the private user files which (for want of a disc) are kept on a magnetic tape. For convenience the format of the private tape files is such that they can be written or punched using the off-line equipment. (This also offers a crude form of large-scale input-output.) Tape files are composed of binary-mode card images, initiated by a Hollerith title card (cols. 1-6 = title, cols. 7-12 = class) and terminated by an end-of-file mark. (The first file should be of class "first", the last file of class "last".)

The background system requires 7 tapes to operate, 3 on channel A and 4 on channel B. For each foreground user, two tapes are used, one as a private file tape and one as a dump tape. These tapes are staggered on channels A and B or channels B and C. The Supervisor and the commands are stored on the TSS System Tape, mounted on C1.

The commands are typed by the user to TSS (not to his own program) and thus can be initiated at any time regardless of the particular user program in memory. For similar coordination reasons, TSS handles all I/O of the foreground system typewriters. Commands are composed of segments separated by vertical strokes; the first segment is the command name and the remaining segments are parameters pertinent to the command. Each segment consists of the last 6 characters typed (starting with an implicit 6 blanks) so that spacing is an easy way to correct a typing mistake. A carriage return is the signal that initiates action on the command. Whenever a command is received by the Supervisor, "WAIT", is typed back followed by "READY." when the command is completed. While typing, an incomplete command line may be ignored by the "quit" sequence of a code delete signal followed by a carriage return. Similarly after a command is initiated, it may be abandoned if a "quit" sequence is given. In addition, during unwanted command typeouts, the command and output may be terminated by pushing the special stop output button.

Commands

In addition to the login, logout, listf, and octal patching and inspecting commands that were checked out, the following commands have been added and are now working correctly:

1. input

Sets user in input mode and initiates automatic generation of line numbers. The user types a "card image" per line according to a format appropriate for the programming language. When in the automatic input mode, the manual mode may be entered by giving an initial carriage return and typing the appropriate line number followed by | and the line, for as many lines as desired. To reenter the automatic mode, an initial carriage return is given. The manual mode allows the user to overwrite previous lines and to insert lines (cf. File Command).

2. edit | α | β

α = title of file, β = class of file

The user is set in the automatic input mode with the designated file as initial input lines. The same conventions apply as to the input command.

3. file | α | β

α = title to be given to file.

β = class of language used during input.

The created file will consist of the numbered input lines in sequence; in the case of duplicate line numbers, the last version will be used. The line numbers will be written as sequence numbers in the corresponding card images of the file.

For convenience the following editing conventions apply to input lines:

a. An underline signifies the deletion of the previous characters of the line.

b. A backspace signifies the deletion of the previous character in the field.

The following formats apply:

a. FAP: symbol, tab, operation, tab, variable field and comment;

b. Mad, MADTRN, FORTRAN: statement label, tab, statement. To place a character in the continuation column: statement label, tab, backspace, character, statement;

c. DATA: cols. 1-72.

4. fap | α | β

Causes file designated as α,fap to be translated by the FAP translator. Files α,symtb and α,β (where β is BSS) are added to the user's private file tape giving the symbol table and the relocatable binary BSS form of the file.

5. mad | α

Causes file α,mad to be translated by the MAD translator. File α,bss is created.

6. load $|\alpha_1|\alpha_2|\cdots|\alpha_n$

Causes the consecutive loading of files α_i,bss (i=1,2,...,n). An exception occurs if α_i = (libe), in which case file α_{i+1},bss is searched as a library file for all subprograms still missing. (There can be further α_i as well as further library files.)

7. use $\alpha_1|\alpha_2|\cdots|\alpha_n$

This command is used whenever a load or previous use command notifies the user of an incomplete set of subprograms. Same α_i conventions as for load.

8. start $|\alpha|\beta$

Starts the program setup by the load and use commands after first positioning the user private file tape in front of the title card for file α,β. (If β is not given, a class of data is assumed; if both α and β are not given, no tape movement occurs and the program is started.) The start command may also be used to restart a dormant program.

9. pm $|\alpha$

α = "lights", "stomap", or usual format of F2PM request: subprogram name $|loc_1|loc_2|$mode$|$direction where mode and direction are optional.

Produces post-mortem of user's dormant programs according to request specified by α.

10. skippm

Used if a pm command is abandoned on output and the previous interrupted program is to be restarted.

11. printf $|\alpha|\beta|\gamma$

Types out file α,β starting at line number γ if γ is given. If γ is not given, typeout begins with the first record. Whenever output buffer fills, the command program goes into I/O wait status allowing other users to time-share until buffer needs refilling.

12. xdmp $|\alpha|\beta$

Creates file α,β (if β omitted, xdump class assumed) on user's private file tape consisting of complete state of user's last dormant program.

13. xundmp | α | β

Inverse of xdmp in that it resets file α,β as user's program, starting it where it last left off.

The command madtrn | α has been written but is not checked out. Madtrn | α will cause the file α|madtrn to be edited into an equivalent file α | mad and translation will then occur as if the command mad | α had been given.

1.2 Real-Time, Time-Shared Computer Project

H.M. Teager
B.W. Munday
M. Kudlick
A. Rutchka
E. Kliman

Progress has continued over the past semester in the three broad areas of hardware, software, and end applications. Each of these will be considered in turn.

I. End Applications

The principal objective of this group is the effective linkage of scientist or engineer with a computer system to handle problem solving in the best achievable manner, using each partner for the operations which best match his capability, and communicating back and forth with the most flexible and meaningful language and devices for presentation of problems and solutions. To make possible the achievement of low-cost real-time interaction, this group has approached the problem from the context of time-sharing a large computer, but should the economics of the situation reverse in favor of smaller machines, the development work of devices, applications, and software would remain valid. The necessary time-sharing techniques are, however, being developed and tested.

From an applications viewpoint, this group has been exploring a variety of end usages, where graphical I-O, including handwritten and sketched information and language forms would be useful. These applications include:

1) Use of a digital computer as an analogue device with hand-sketched input, to replace, for example, electrolytic tanks in microwave plasma research and special purpose electrode design.

2) Use for lumped and distributed, nonlinear, active network analysis and design.

3) Use for logical circuit analysis and design.

4) Use of the complete set of ordinary mathematical symbolism (superscripts, subscripts, sigma, parenthesis free, special Greek symbols: π, $\in$, etc.) in flow chart forms of ordinary programs.

5) Algebraic series and polynomial manipulation, for approximation and solution of nonlinear integral and differential equations problems.

As each area is explored, input and output languages are being devised to describe the most general types of operations desired, as well as techniques to transform these "problem-oriented" languages into an algebraic, procedure-oriented common language. Most of this work is largely in the form of individual thesis research, and as of yet is largely in the exploration and programming stage. However, it is extremely clear even now that a properly integrated system of graphical input and output capability makes feasible an immense, yet untapped, potential for man-machine cooperation.

II. Hardware

The major hardware effort of the group has been the design and construction of a special purpose satellite computer, under the direction of Mr. Peter Jensen. This computer, which is now in the final stages of checkout, is fully capable of providing the necessary information and control signals for up to 10 calcomp 560-R digital plotters. The computer accepts information from the 7090 computer in the form of a machine word for line segment length and slope, and acts as a buffer store for up to 64 line segments for each plotter. The design of the computer is currently being modified to allow the inclusion of rate of change of slope information, in order to allow the drawing of curved line segments with a single word. Modifications to the present buffer between the 7090 data channel and the I-O equipment are now almost complete. The buffer now has the capability to control three Flexowriter typewriters, the plotter computer and a photoelectric tape reader.

Evaluation of a low-cost handwriting and graphical input device under development outside this project is continuing and plans including software are being made for its inclusion in the console equipment. For medium data-rate and low-accuracy graphical display work, a low-cost commercial device is currently being evaluated. This device has storage for up to 500 matrix dot characters on its small CRT face, and by a suitable change could be modified to use characters as small line segments. It is hoped that this device, including its own keyboard, will eventually supercede the present typewriter portion of the I-O system.

Work is continuing on the development of larger-scale, classroom I-O facilities, including blackboard display, and graphical input. At

present, it would appear that electroluminescent devices for display, as well as scaled up graphical inputs and page projector equipment may prove adequate for this purpose.

III. Software

Each device and end usage that is developed raises its own software problems, over and above the general programming system problems for time-sharing per se. For this reason, the necessary software work has been proceeding in parallel with each hardware development.

For example, the plotter capability is being supported by development of computer subroutines for drawing labeled plots of bar charts, linear, semi-log, and log-log two-dimensional functions, and isometric and contour map projections of higher dimensional functions. Modifications are also being made in an algebraic compiler, including its source language, so that these functions can be easily and unambiguously requested. Similarly, the graphical and handwritten input capability is being supported by the development of small, fast, reliable routines for decoding such information, utilizing known, analyzed samples of the particular user's character set.

The major software task of the bulk of the programming group, however, is to complete the internal modifications to the MAD compiler, principally a complete reorganization of internal tables and memory usage, in order to achieve the following objectives:

1) Reduce the compiler's storage requirements at any time to less than 10,000 registers.

2) Allow program debugging, correction, and modification, in a language on the same level as the source language, at run or compile time without time loss penalties.

3) Design the compiler to make full use of the real-time mode of usage.

4) Build in added capability, such as complex number, double precision, polynomial, and algebraic power series manipulations.

All of these objectives, particularly the first three are well under way, and have been materially assisted by the existence of the "program analysis" program developed by this group.

Work is also continuing on the scheduling, I-O control, and monitor program that is to be used for the prototype time-sharing system. Work on the information retrieval portion of this system (fetching and dispatching of data and programs to and from disc files) is still largely on an ad-hoc cut and try basis. More basic theoretical work will be done

in this area in the near future.

The development of software support for problem-oriented languages is, as of yet, largely in the stage of thesis investigations, rather than under active development of production type programs. It is hoped that by June the thesis research work will make clearer the most suitable forms of such problem-oriented languages. The basic decision, however, to design all special purpose computers for problem-oriented languages to produce algebraic language "object" programs (which can then go through a second compilation process) remains unchanged.

1.3 Artificial Intelligence Project

J. McCarthy
M.L. Minsky

1.3.1 LISP

The LISP system for computing with symbolic expressions was further developed. A new compiler and a new manual are in the works.

1.3.2 Kalah

A program for playing "kalah" was developed. The major advance over previous game-playing programs is the α-β heuristic that, combined with a good plausible move generator, permits a look-ahead of double the depth of the simple minimax scheme. In the case of "kalah" a look-ahead of 14 levels from the initial position is feasible. Selected variations are printed with the values computed for each and the amount of the effort put into computing the value. This gives much useful information on what the machine spends its time thinking about.

1.3.3 Theorem Proving

A LISP program for the Beth method of semantic tableaux for proving theorems in the predicate calculus has been checked out. The method has been extended to the predicate calculus with equality.

1.3.4 Theory

Theoretical studies of the mathematical theory of computation, theory of computability, and artificial intelligence have been continued. McCarthy presented a paper on Mathematical Theory of Computation at the Conference on Mathematical Logic and Non-Numerical Computation held at Blaricum, Holland.

The scheduling program is one important link in the over-all time-sharing control system. Another, concerning the control of direct-data input-output operations, has been successfully checked out during the past half year. In that program, which also conforms to the space and time requirements mentioned above, input and output is handled by the data channel with the CPU functioning only as an overseer to initiate channel action and decide upon further action when the channel has disconnected. In this way, there is no waiting by the CPU for direct-data input-output operations; the intent is always to have one of the time-shared programs running during such operations. In addition to these programs, I have devised a few diagnostic routines to test the external hardware used in conjunction with this project. One program was used on the 709 to test the signalling and response of the electric typewriters and photoelectric tape reader via the direct-data channel, and provided a means for debugging the electronic equipment associated with these external units. A program now being prepared for the 7090 will measure the data rates of these units.

2.1.2 Progress Report of Mr. Paul W. Abrahams, Department of Mathematics, MIT

As of last June, the LISP programming system had a compiler associated with it. However, the compiler was rather inflexible and difficult to modify when changes and additions were introduced into LISP. It did not permit communication between compiled programs and interpreted programs except in very limited ways, and for a number of users, myself included, this was a considerable disadvantage. The final blow was dealt to this compiler when its author, Robert Brayton, left the Institute. Since the program was virtually uncommented, there was little hope of resurrecting it, and it was removed from the operating system. But despite its drawbacks, it had proved its point. A LISP compiler could be built which would increase the operating speed of LISP programs by a factor of 60 over interpretation.

During this past summer, I worked with M.D. McIlroy at Bell Telephone Laboratories on a LISP compiler which operated within Bell-FAP. The Bell compiler had many desirable features, but it nevertheless was not suitable for use at MIT. However, as a result of my experience I decided to program a compiler which would fulfill the requirements of the MIT LISP system and nevertheless retain the advantages of the Bell compiler. This has been my major project under my assistantship this term. The compiler has been programmed entirely in LISP; one consequence of this is that it is quite modular in construction, and pieces of it may be changed to meet future changes in the system. The philosophy has been to ask the programmer to declare information rather than force the compiler to do an inadequate and costly job of ferreting it out. Most of this information relates to the communication of variables. There are various ways in which communication may take place: communication of arguments from a function to a subfunction; communication of free variables among a set of compiled functions;

and communication of variables between compiled functions and interpreted ones. Information is not always available from analysis of the programs to be compiled as to which types of communication will be needed; and the different types differ markedly in cost. Therefore, the simplest and most frequent case, viz., communication of arguments between compiled functions, is assumed unless a different situation is declared. This saves effort on the part of the compiler while at the same time giving the programmer more control over the sort of compiled program he gets. Furthermore, unnecessary restrictions on the order of compilation are eliminated, since the compiler no longer needs to draw conclusions from this order. This approach is useful (though somewhat controversial) for compiler-writing in general; for instance, it has been suggested by Alan Perlis, among others, that ALGOL be changed so that arrays be assumed to have fixed dimensions unless declared to be variable; and functions be assumed to be non-recursive unless declared to the contrary. Thus the most expensive general case is assumed only when specifically requested.

The programming of the compiler in LISP has been completed; however, much work remains to be done before it can be made an integral part of the LISP system. Theoretically, the compiler could be interpreted and then would compile itself. The main difficulty lies with the form of the output, which is dependent on certain absolute locations within LISP; and these locations change from one assembly to the next. However, I may be able to modify the compiler to circumvent this. It would be a useful result to be able to avoid the need for hand-coding the compiler at all.

In addition to my work on the LISP compiler, I have acted as chairman of the Computation Center Seminar. This has involved locating speakers and arranging publicity for them. There have been talks on a wide variety of topics this year. Subjects have ranged through time-sharing, computer organization, use of a computer to generate complicated music (complete with tape-recorded demonstration), artificial intelligence, and numerical analysis.

2.1.3 Progress Report of Mr. John L. Rhodes, Department of Mathematics, MIT

The problem was to study "non-probabilistic discrete unit" machines. The attack was broken into two parts: first to reduce problems concerning such machines to algebraic problems, and second to use algebraic techniques to solve the corresponding algebraic problems. The major problem was: given a function f, find those (small) machines such that (high) iterates yield an input-output function which is "more capable" than f. Here f "more capable than" g means $qfp=g$ where q and p are restricted to be a simple type of function. This problem was reduced to algebra by assigning to each machine a semi-group such that the input-output functions of a machine M are more capable than those of a machine N if and only if the semi-group of N is a homomorphic image of a sub-semi-group of the semi-group for M.

TABLE III

REVISED ABSTRACTS

M1371 THE INCEPTION, BIRTH, AND GROWTH OF NEW FIRMS
Richard F. Miller, Jr., School of Industrial Management

The study is being continued, but has been combined with the work being done under Problem Number M1672. The majority of the time for the past quarter has been spent testing a revised model now being run under M1672.

The previous abstract is still valid, with one revision. The scope of the project has been expanded to include a proposed case study, in addition to the original hypothetical model.

M1416 AN EXPERIMENTAL TIME-SHARING SYSTEM
Dr. F. J. Corbató, Computation Center

An experimental time-sharing system has been designed for use with the 709 computer, two flexowriter typewriters and the real-time equipment rack designed and built under the direction of H. Teager and his group. The time-sharing occurs between three users, two on-line each at a typewriter in a foreground system, and a third passive user of the background. Fap-Mad-Madtran BSS Monitor System (see memo CC-177) similar to the FMS Monitor System used by most of the Center programmers.

Significant features of the foreground system are that it allows the user to: (1) develop programs in languages compatible with the background system, (2) develop a private file of programs, (3) start debugging sessions at the state of the previous session, and (4) set his own pace with little waste of computer time. It is organized around commands that the user can give on his typewriter and the private user files which (for want of a disc) are kept on a magnetic tape. For convenience, the format of the private tape files is such that they can be written or punched using the off-line equipment. Core storage is allocated such that all users operate in the upper 27K with the time-sharing supervisor (TSS) permanently in the lower 5K. To avoid memory allocation clashes, protect users from one another, and simplify the initial system organization, only one user has been kept in core memory at a time. User swaps are kept down to overlapped magnetic tape dump-undumps of the pertinent locations in the two user programs.

N1599 STATISTICAL ANALYSIS OF THE COSTS OF COMMERCIAL BANKING
John F. Zoellner, Harvard University

The computational problem is a statistical analysis of the costs and earnings of commercial banking. The results of the statistical analysis will be the core of an industry study of commercial banking. In particular, the study will be concerned with government regulation of commercial banking.

The statistical analysis consists of three parts. First, cost functions will be fitted to different categories of banking costs. Second, earnings functions will be fitted to different categories of earnings. Third, an analysis will be made of the nature and adequacy of bank profits. Two sets of data will be analyzed, each set encompassing four years of experience. One set of data has eighty-five observations on seventy-seven variables. The second set has 295 observations on sixty-three variables.

VI. PUBLICATIONS OF THE CENTER

6.0 Computation Center Memoranda

Computation Center Memoranda are identified by the letter designation CC. Although certain Computation Center Memoranda are for internal circulation only, there are others which are of specific interest to all 709 programmers at the time of printing. The following are available on request as long as the supply lasts.

CC-173 Linking Segment System Description and Use Within the FMS System, J. McCarthy, F.J. Corbato, M.L. Merwin, and R. Daley, February 16, 1961 (8 pages).

CC-174 Description of a 709 FAP Version of XSIMEQF, XDETRMF, A.M. Neill, February 27, 1961 (6 pages).

CC-176 Octal Correction Cards Within the FMS System, R. Daley, May 15, 1961 (2 pages).

CC-178 Availability of COMIT, V. Yngve, June 22, 1961 (2 pages).

CC-181-1 The Use of MAD in the MIT Version of the FMS System, L. Korn and M. Merwin, December 13, 1961 (3 pages).

CC-182 Common Programming Errors, Consulting Staff, September, 1961 (5 pages).

CC-183 General Description and Instructions for the Use of the Time-Sharing Console, M. Kovarik and A. Rutchka, October 4, 1961 (7 pages).

CC-185 An Experimental Time-Sharing System for the 709, F.J. Corbató, M.L. Merwin, R. Daley, November 28, 1961 (5 pages).

CC-186 Fortran and MAD Format Specifications, J. Spall, November 1961 (10 pages).

CC-187 709-7090 Conversion, Computation Center Staff, December 4, 1961 (3 pages).

CC-188 Madtran - A Fortran-to-MAD Language Translator, L. Korn, December 11, 1961 (3 pages).

CC-189 Latest Writeup on MIT 7090 RPQ for Relocation and Protection Modes, M.L. Merwin, December 12, 1961 (4 pages).

6.1 Professional Society Papers; Seminars

During the past six months, the following papers and seminars were presented by members of the Computation Center staff.

"An Experimental Time-Sharing System for the 709 Computer", Dr. F.J. Corbató, presented at the University Directors' Conference, Endicott, N.Y., July 19, 1961. (To be published in the Proceedings.)

"Trends of Computer Usage", Dr. F.J. Corbató, at the dedication of the University of New Hampshire Computation Center, December 2, 1961.

"Computers and the High School", Dr. F.J. Corbató, to the Association of Teachers of Mathematics in New England, October 21, 1961.

"Cauchy's Method of Minimization", Dr. A.A. Goldstein, Mathematics Colloquium at Iowa State University, October 1961.

"Mathematical Theory of Computation", given by Prof. J. McCarthy at a Conference on Mathematical Logic and Non-Numerical Computation held at Blaricum, Holland.

"Computers in Universities", a talk given by Prof. P.M. Morse at the opening ceremonies of the new Computation Center at the University of Massachusetts, September 23, 1961.

"The Use of Computers in Operations Research", Prof. P.M. Morse, the first in a series of invited talks at the new Computation Center of Brown University, October 3, 1961.

"Systems Considerations in Real-Time Computer Usage", paper presented by Prof. H.M. Teager at the ONR Symposium of Automated Teaching, October 1961 (to be published as a book with the other papers and panel discussions).

"The Computer of the Future", Prof. H.M. Teager, given to the MIT Club of Worcester, Mass., October 19, 1961.

6.2 Publications

Members of the staff of the Computation Center have published the following during the last six months.

Dr. F.J. Corbató:

Reviewed "Electronic Digital Computers" by Franz L. Alt in Computing Reviews, Vol. 2, #4, July-August 1961.